2003

Reversible Logic Synthesis with Cascades of New Family of Gates

Mozammel H.A. Khan
Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology

Marek Perkowski
Portland State University, marek.perkowski@pdx.edu

Reversible Logic Synthesis with Cascades of New Family of Gates

Mozammel H. A. Khan and Marek A. Perkowski*
Computer Science and Engineering Department, East West University, Dhaka 1212, Bangladesh.
mhakhan@ewubd.edu
Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Gusong-dong, Yusong-gu, Daejeon 305-701, KOREA. mperkows@ee.kaist.ac.kr

Abstract: Reversible circuits are currently on of top approaches to power minimization and the one whose importance will be only growing with time. In this paper, the well known Feynman gate is generalized to *k*k* gate and a new generalized *k*k* family of reversible gates is proposed. A synthesis method for multi-output SOP function using cascades of the new gate family is presented. For utilizing the benefit of product sharing among the output functions, two graph-based data structures are used. Another synthesis method for AND-OR-EXOR function using cascades of the new gate family and generalized Feynman gate is also presented. Synthesis method for single-output ESOP function using cascades of the new gate family is also presented. All these synthesis methods are technology independent and generate very few garbage outputs and need few input constants.

1. Introduction

Motivation to Study Reversible Logic

Landauer [1] showed that a computational system built using traditional irreversible logic gates such as AND or multiplexer leads inevitably to energy dissipation, regardless of the technology used to realize the gates. The energy loss due to irreversible gates is negligible for current silicon technologies using adiabatic design. However, it is well known that Moore's Law will stop to function around years 2010 – 2020 and some dramatic changes will therefore have to happen in microelectronics not latter than the middle of this century [2]. It this time reversible design will become of primary importance.

Bennett [3] showed that for power not to be dissipated in an arbitrary circuit, it is necessary that the circuit be built from ***reversible gates***. In principle, reversible logic gates dissipate arbitrary little heat and the use of reversible operations is likely to become more attractive. It should be noted that Bennett's theorem is only a necessary but not sufficient condition. Its extreme importance lies in the technological necessity that ***every future technology will have to use reversible gates in order to reduce power***. Quantum logic is reversible and the problem of efficient designs of quantum circuits [4] includes as its sub-problem the problem of synthesis using classical reversible gates. Therefore, many of the methods for reversible logic synthesis can be adapted to quantum logic as well.

Reversible Logic

Reversible logic are circuits (gates) that have the same number of inputs and outputs and have one-to-one mappings between input vectors and output vectors; thus the input vector states can be always uniquely reconstructed from the output vector states. A reversible gate with *k* inputs and *k* outputs is called a *k*k* gate. All gates in a reversible circuit must be reversible. A ***balanced function*** has half of minterms with value 1 and half with value 0. A reversible circuit without constants on inputs realizes a balanced function on each of its outputs. Therefore, it can realize non-balanced functions only with ***garbage*** (*unused*) outputs and constant values on its primary inputs.

As the truly low-power circuits (with power arbitrarily small) cannot be built without the concept of reversible logic, various technologies for reversible logic are recently intensively studied. These technologies include: (i) standard CMOS [5, 6], (ii) optical technologies [7, 8], (iii) quantum logic technologies [9, 10], (iv) DNA technology, and (v) mechanical technology (nanotechnology) [11].

Many binary universal reversible logic gates have been proposed [5, 12-15]. There exists only one 1*1 gate, which is an ***inverter*** (a wire is not considered as a gate). This gate is shown in Figure 1(a) and is very important since it does not introduce garbage outputs. There are several 2*2 gates in reversible logic and they are all linear. A gate is ***linear*** when all its outputs are linear functions of input variables. The 2*2 ***Feynman gate*** (also called ***controlled-not*** or ***quantum EXOR*** or ***reversible EXOR***) is shown in Figure 1(b). This gate is a ***one-through gate***, which means that one of its input variables is also an output. When $A = 0$ then $Q = B$, when $A = 1$ then $Q = B'$. With $B = 0$ the 2*2 Feynman gate is used as a ***fan-out*** (or ***copying***) ***gate***. Every linear reversible function can be built by using only 2*2 Feynman gates and inverters. There exist 8! = 40,320 3*3 reversible logic gates, some of them with interesting properties. Two of the universal 3*3 reversible gates have been much studied: ***Fredkin gate*** [12] and ***Toffoli gate*** (also called ***3*3 Feynman gate*** or ***Controlled-***

controlled-not). The 3*3 *Fredkin gate* is shown in Figure 1(c). In terms of classical logic, this gate is just two multiplexers controlled in a flipped (permuted) way from the same control input *A*. Fredkin gate is also a one-through gate. The 3*3 *Toffoli gate* is shown in Figure 1(d). Toffoli gate is a *two-through gate*, because two of its inputs are returned unmodified as its outputs. When $C = 0$ then $R = AB$, so AND gate is realized on *R* output. When $A = 1$ then $R = B \oplus C$, so EXOR gate is realized on *R* output. When $B = 1$ then $R = A \oplus C$, so again EXOR gate is realized on *R* output. The 3*3 ***Kerntopf gate*** [13, 16] is shown in Figure 1(e). When $C = 1$ then $P = A + B$, $Q = AB$, $R = B'$, so OR and AND gates are realized on outputs *P* and *Q*, respectively, with *C* as the controlling input value. When $C = 0$ then $P = A'B'$, $Q = A + B'$, $R = A \oplus B$. Therefore, for control input value 0, the gate realizes NOR, IMPLICATION, and EXOR on its outputs *P*, *Q*, and *R*, respectively. The 3*3 Kerntopf gate is not a one-through nor a two-through gate. Generalized *n*n* Fredkin, Toffoli, and Kerntopf gates are introduced in [17].

Reversible Logic Synthesis

Reversible logic synthesis is not as easy as classical logic synthesis. The major constraints of reversible logic synthesis are: (i) the fan-out of every signal, including primary inputs, is one, (ii) the graph of the reversible circuit must be a dag (directed acyclic graph), which means that there must be no loops of gates or internal loops in a gate, and (iii) many of the practical functions are not themselves reversible and need to make reversible before implementing them with reversible gates. Systematic logic synthesis algorithms for reversible logic are still very immature, but some methods have been proposed [15-21]. Most of the papers discuss design using Feynman, Toffoli, and Fredkin gates. In [20] compositional synthesis methods for reversible logic have been presented. The simplest structure for composition is cascades. Reversible logic synthesis using cascades of gates is presented in [15, 17, 21]. The cascades have the same number of intermediate signals at every level. When a gate is applied for composition it subtracts as many input variables as it has inputs and it adds the same number of new variables to the next level. The reversible cascades are now very popular, as many well-known standard logic methods can be adapted to reversible cascades.

The objectives of a good synthesis algorithm for reversible logic are: (i) not to create excessive garbage outputs, (ii) not to create unnecessary input constants, (iii) to avoid leading output signals of gates to more than one input, as such fan-out requires additional copying gate, and (iv) to minimize the length of the circuit to reduce the delay of the circuit.

Contribution of This Paper

The goal of this paper is to develop efficient logic synthesis methods based on cascades of new reversible gate families and new design algorithms. For this purpose, new family of reversible gate is proposed. For the first time, to our knowledge, synthesis method for multi-output SOP and AND-OR-EXOR functions are presented in this paper. Moreover, synthesis method for single-output ESOP function is also presented. For all these methods two graph-based data structures are used. The developed methods are very efficient in the sense that they produce very few garbage outputs and need few input constants.

2. New Families of *k*k* Reversible Gates

A *k*k* Feynman gate is proposed in Figure 2(a). From the truth table it can be shown that the gate is a reversible gate. It is a $(k - 1)$- through gate and the output P_k produces the EXOR-sum of the input variables.

A new generalized *k*k* reversible gate family is proposed in Figure 2(b), where f_{k-2} is an arbitrary function of $A_1, A_2, \cdots, A_{k-2}$ and f'_{k-2} is the complement of the earlier function. From the truth table it can be shown that the gate is a reversible gate. The gate is a $(k - 2)$-through gate. Depending on f_{k-2}, many possible gates can be constructed. When $A_{k-1} = 0$ then $P_{k-1} = A_k$ and $P_k = f_{k-2} + A_k$, so A_k is copied on P_{k-1} output and OR-sum of f_{k-2} and A_k is given on P_k output. When $A_{k-1} = 1$ then $P_{k-1} = f_{k-2} \oplus A_k$ and $P_k = (f_{k-2} + A_k)'$, so EXOR and NOR operations are realized on P_{k-1} and P_k outputs, respectively. When $A_k = 0$ then $P_{k-1} = f_{k-2} A_{k-1}$ and $P_k = f_{k-2} \oplus A_{k-1}$, so AND and EXOR operations are realized on P_{k-1} and P_k outputs, respectively. When $A_k = 1$ then $P_{k-1} = (f_{k-2} A_{k-1})'$ and $P_k = A'_{k-1}$, so NAND and NOT operations are realized on P_{k-1} and P_k outputs, respectively.

3. Synthesis of Multi-Output SOP Function

For multi-output SOP realization using the new *k***k* gate, we assume that $f_{k-2} = \prod_{\exists i \in \{1,2,\cdots,k-2\}} A_i$, that is, f_{k-2} is a product of some of the input variables $A_1, A_2, \cdots, A_{k-2}$. For realizing a SOP, we will always use $A_{k-1} = 0$ such that $P_{k-1} = A_k$ and $P_k = f_{k-2} + A_k$.

In a multi-output SOP function, some sub-expressions may be common among more than one output function. Getting advantage of such a common sub-expression can not be handled in a method similar to that used in circuits with classical gates. The main constraint is that the fan-out of all signals in a reversible gate is one. Therefore, this problem has to be handled in a different way to achieve maximum benefit.

The method for synthesis of multi-output SOP function using the new gate family is illustrated using the example function: $F_1 = BC' + AB' + A'B'C$, $F_2 = BC' + AB' + A'BC$, and $F_3 = AB' + A'BC + AC'$.

Step 1. The product sharing information is summarized in Table 1. The table shows the occurrence of all product terms in different output functions and the total number of occurrence of each product term.

Step 2. Depending on the information of the product sharing table, the connectivity tree (or forest of shared trees) for the product terms is constructed as shown in Figure 3(a). In the connectivity tree each node represents a product term. The product term(s) with the highest no of occurrence is placed at the top level. Then the product term(s) with decreasing order of no of occurrence is placed below each level other. The nodes are connected by directed edges so that ORing the products in a path forms a function. For example, $AB' + BC' + A'B'C$ forms the function F_1. A node must not have more than one inward edge and if needed a separate node is created for the product term. For example, two nodes are created for the product term $A'BC$. Though the fan-out of a signal is limited to one, multiple outward edge from a node will be allowed and this will be handled in a very efficient way.

Step 3. From the connectivity tree, the implementation tree is created as shown in Figure 3(b). In this tree a node represents a new *k***k* gate. As the A_{k-1} input of the gate is always 0, it is not explicitly shown. The inward edge of the node represents the A_k input. The left child represents the $P_k = f_{k-2} + A_k$ output and the right child represents the $P_{k-1} = A_k$ output. The product inside the node represents the function f_{k-2}. In the connectivity tree, some nodes have multiple outward edges requiring multiple fan-out. It is clear that the right child of an implementation node copies the input value and thus provide the fan-out of the input value. This phenomenon is used to manage multiple fan-out requirement of the connectivity tree.

Step 4. The implementation tree is realized by cascades of new *k***k* gates as shown in Figure 3(c). For shortening the cascade length (and thus decreasing the circuit delay), all long branches of the implementation tree are realized by separate parallel cascades. For this purpose, the primary inputs are copied using 2*2 Feynman gates with $B = 0$. Also note that the 0 at the P_{k-1} output of the first gate is connected to the A_{k-1} input of the next gate to reduce the input constants. In this particular example, only three garbage outputs are generated and six input constants are needed.

4. Synthesis of AND-OR-EXOR Function

In many cases AND-OR-EXOR realization of a function requires fewer gates that ESOP realization [22]. For realizing AND-OR-EXOR function, the SOP functions are realized using the method of Section 3. Then the SOP outputs are EXOR-summed using a *k***k* Feynman gate. Realization of an arbitrary AND-OR-EXOR function $F = (AC' + A'C) \oplus (AB + A'B)$ is shown in Figure 4.

5. Synthesis of Single-Output ESOP Function

For single-output ESOP realization using the new *k***k* gate, we again assume that the function f_{k-2} is a product of some of the input variables $A_1, A_2, \cdots, A_{k-2}$. For this realization we will always use $A_k = 0$ such that $P_{k-1} = f_{k-2}A_{k-1}$ and $P_k = f_{k-2} \oplus A_{k-1}$. The synthesis method for single-output ESOP function using the new *k***k* gate is illustrated using the example function $F = BC' \oplus AB' \oplus A'B'C$.

Step 1. The implementation tree is created as shown in Figure 5(a). A node of the tree represents a new *k***k* gate. As the A_k input of the gate is always 0, it is not explicitly shown. The inward edge of the node represents the A_{k-1} input. The left child represents the $P_k = f_{k-2} \oplus A_{k-1}$ output and the right child represents the $P_{k-1} = f_{k-2}A_{k-1}$ output. The product inside the

node represents the function f_{k-2}. Some of the P_{k-1} outputs may generate 0s. These 0s are identified and indicated in the tree. Moreover, the ordering of the products in the tree is rearranged to increase such 0 output. Each such 0 is connected to the A_k input of the next gate of the cascade to reduce the garbage outputs as well as input constants.

Step 2. The implementation tree is realized by cascades of new *k***k* gates as shown in Figure 5(b). In this particular example, only one garbage output is generated and two input constants are needed.

6. Experimentation

A preliminary version of the program is developed in C language. In this implementation, it is assumed that the functions are already available as SOP/ESOP form. All the examples of this paper are generated using this program. The program is now being enhanced using advanced data structures for function representation like decision diagrams.

7. Conclusion

We generalized the well-known Feynman gate to *k***k* gate and proposed a new generalized *k***k* reversible gate family. To our knowledge, no method is yet reported for synthesizing multi-output SOP and AND-OR-EXOR functions using reversible gates. In this paper, we presented a synthesis method for multi-output SOP function using cascades of the new *k***k* gate family. For utilizing the benefit of product sharing among output functions, two graph-based data structures – connectivity tree and implementation tree are introduced. We also presented a synthesis method for AND-OR-EXOR function using cascades of the new *k***k* gates and a *k***k* Feynman gate. Another synthesis method for single-output ESOP function using the new *k***k* gates is also presented, where judicious rearrangement of the products in the implementation tree reduces the garbage output as well as the input constants. All these synthesis methods generate very few garbage outputs and need few input constants. Moreover, the methods presented in this paper is technology independent and can be used in association with any known or future reversible technology.

The future research include (i) improving the program using advanced data structure for function representation like decision diagrams, (ii) developing synthesis method for multi-output ESOP function using generalized *k***k* reversible gate family, and (iii) testing the developed method for large benchmark functions.

References

[1] R. Landauer, Irreversibility and heat generation in the computational process. *IBM Journal of Research and Development*, 5, pp. 183-191, 1961.

[2] J. Birnbaum. Computing alternatives. *Talk given at ACM97*, March 3, 1997, San Jose, California.

[3] C. Bennett, Logical reversibility of computation. *IBM Journal of Research and Development*, 17, pp. 525-532, 1973.

[4] D. Deutsch, Quantum computational networks. *Proc. Roy. Soc. Lond*. A 425, 73, 1989.

[5] A. De Vos, Towards reversible digital computers. *Proceedings of European Conference on Circuit Theory and Design*, Budapest, pp. 923-931, 1997.

[6] B. Desoete, A. De Vos, M. Sibinski, and T. Widerski, Feynman's reversible logic gates implemented in silicon. *Proceedings of 6th International Conference MIXDES*, pp. 496-502, 1999.

[7] P. Picton, Optoelectronic, multivalued, conservative logic. *International Journal of Optical Computing*, 2, pp. 19-29, 1991.

[8] P. Picton, A universal architecture for multiple-valued reversible logic. *MVL Journal*, 5, pp. 27-37, 2000.

[9] J. A. Smolin, and D. P. DiVincenzo, Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate. *Physical Review A*, 53, pp. 2855-2856.

[10] A. Peres, Reversible logic and quantum computers, *Physical Review A*, 32, pp. 3266-3276, 1985.

[11] R. C. Merkle, Two types of mechanical reversible logic. *Nanotechnology*, 4, pp. 114-131, 1993.

[12] E. Fredkin and T. Toffoli, Conservative logic. *International Journal of Theoretical Physics*, 21, pp. 219-253, 1982.

[13] P. Kerntopf, A comparison of logical efficiency of reversible and conventional gates. *Proceedings of 9th IEEE Workshop on Logic Synthesis*, pp. 261-269, 2000.

[14] P. Kerntopf, Maximally efficient binary and multi-valued reversible gates. *Proceedings of ULSI Workshop*, Warsaw, Poland, May 2001.

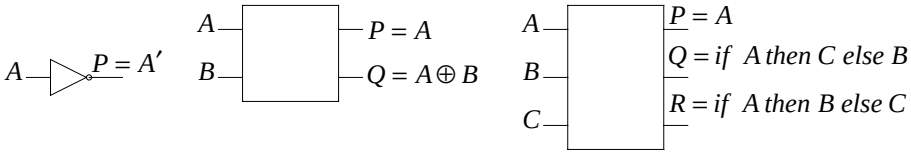
[15] L. Storme, A. De Vos, and G. Jacobs, Group theoretical aspects of reversible logic gates. *Journal of Universal Computer Science*, 5, pp. 307-321, 1999.

[16] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, Regularity and symmetry as a base for efficient realization of reversible logic circuits. *Proceedings of IWLS 2001*, pp. 90-95, 2001.

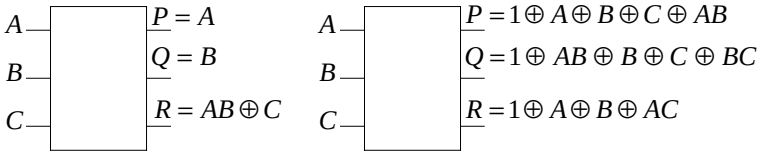
[17] A Khlnopotine, M. Perkowski, and P. Kerntopf, Reversible logic synthesis by gate composition. *Proceedings of IWLS* 2002. pp. 261 – 266.

[18] M. Perkowski, A. Al-Rabadi, P. Kerntopf, A. Mishchenko, M. Chrzanowska-Jeske, Three-dimensional realization of multi-valued functions using reversible logic. *Proceedings of ULSI 2001 Workshop*, Warsaw, Poland, May 2001.
[19] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, Regular realization of symmetric functions using reversible logic. *Proceedings of Euro-Micro 2001*.
[20] M. Perkowski, L. Jozwiak, P. Kerntopf, A. Mishchenko, A. Al-Rabadi, A. Coppola, A. Buller, X. Song, M. M. H. A. Khan, S. Yanushkevich, V. Shmerko, and M. Chrzanowska-Jeske, A general decomposition for reversible logic. *Proceedings of RM 2001*.
[21] A. Mishchenko and M. Perkowski, Reversible Maitra cascades for single-output functions. *Proceedings of IWLS* 2002.
[22] D. Debnath and T. Sasao, A heuristic algorithm to design AND-OR-EXOR three-level networks. Proceedings of ASP-DAC'98, Yokohama, Japan, 1998.

Figures and Tables

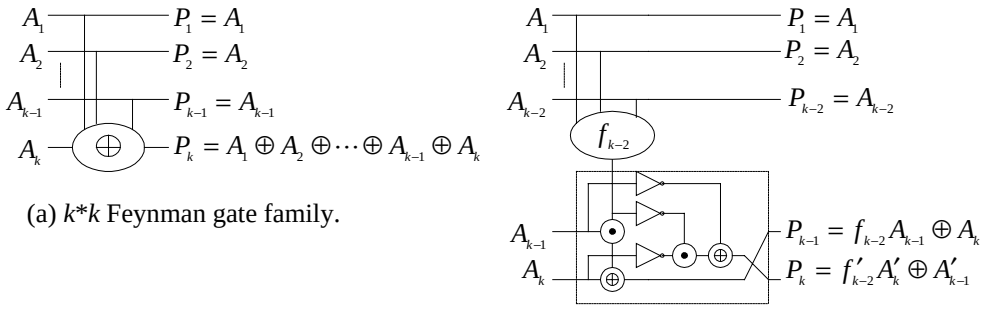


(a) 1*1 NOT (b) 2*2 Feynman gate (c) 3*3 Fredkin gate



(d) 3*3 Toffoli gate (e) 3*3 Kerntopf gate

Figure 1. Some useful binary reversible gates.



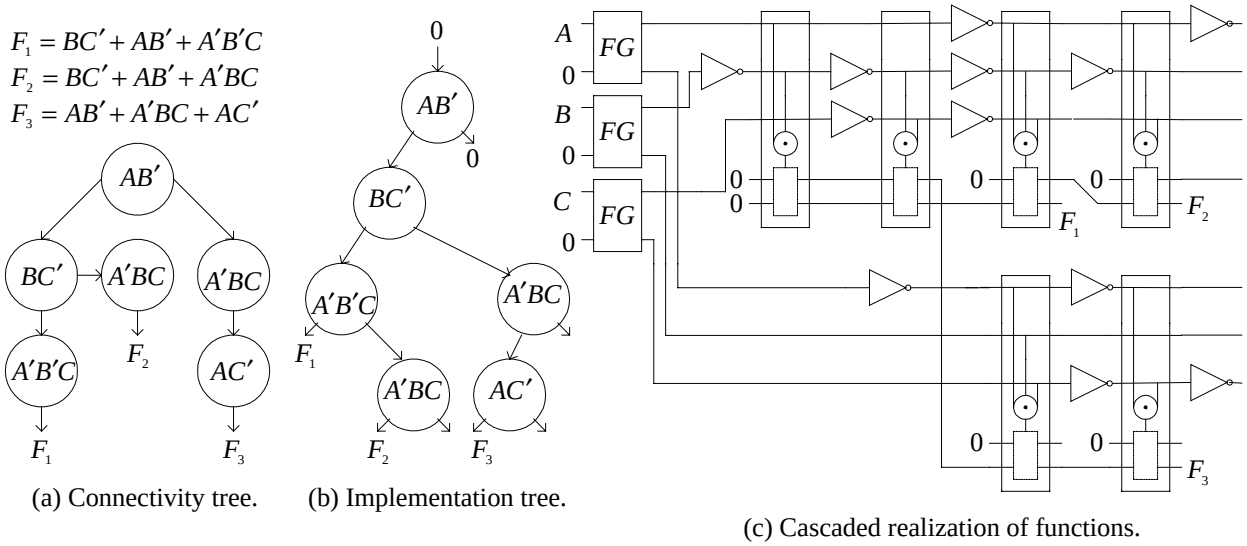
(a) *k***k* Feynman gate family.

(b) A new generalized *k***k* reversible gate family.

Figure 2. New families of *k***k* reversible gates.

Table 1. Product sharing information.

Function Product	F_1	F_2	F_3	No of Occurrence
BC'	X	X		2
AB'	X	X	X	3
$A'B'C$	X			1
$A'BC$		X	X	2
AC'			X	1



(a) Connectivity tree. (b) Implementation tree. (c) Cascaded realization of functions.

Figure 3. Different phases of synthesis of multi-output SOP function.

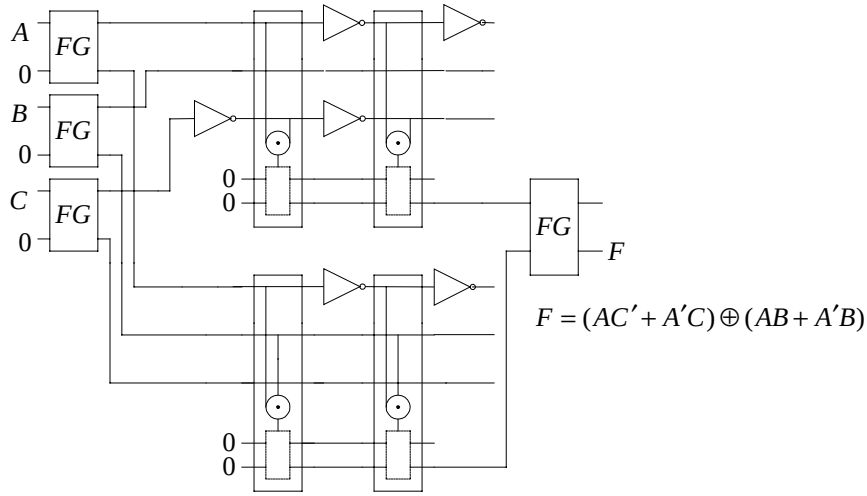
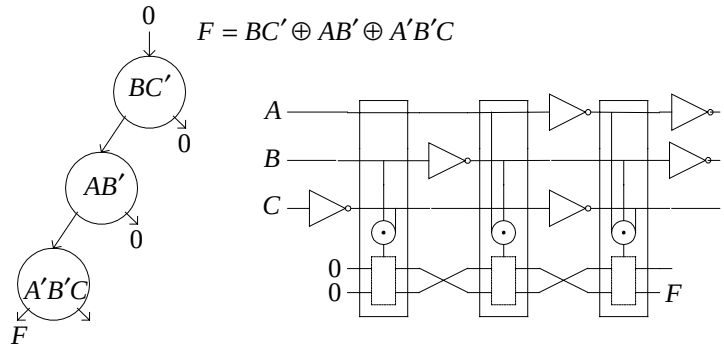


Figure 4. Synthesis of AND-OR-EXOR function.



(a) Implementation tree. (b) Cascaded realization

Figure 5. Synthesis of Single-output ESOP function.